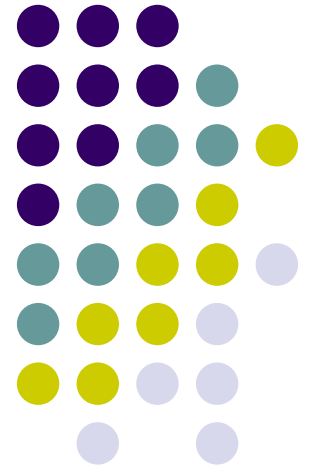


4

Proses



Proses



- Konsep Proses
- Penjadualan Eksekusi Proses
- Operasi pada Proses
- Proses yang saling Bekerjasama (Cooperating Processes)
- Komunikasi Antar Proses (Interprocess Communication)
- Komunikasi pada Sistem Client-Server

Konsep Proses



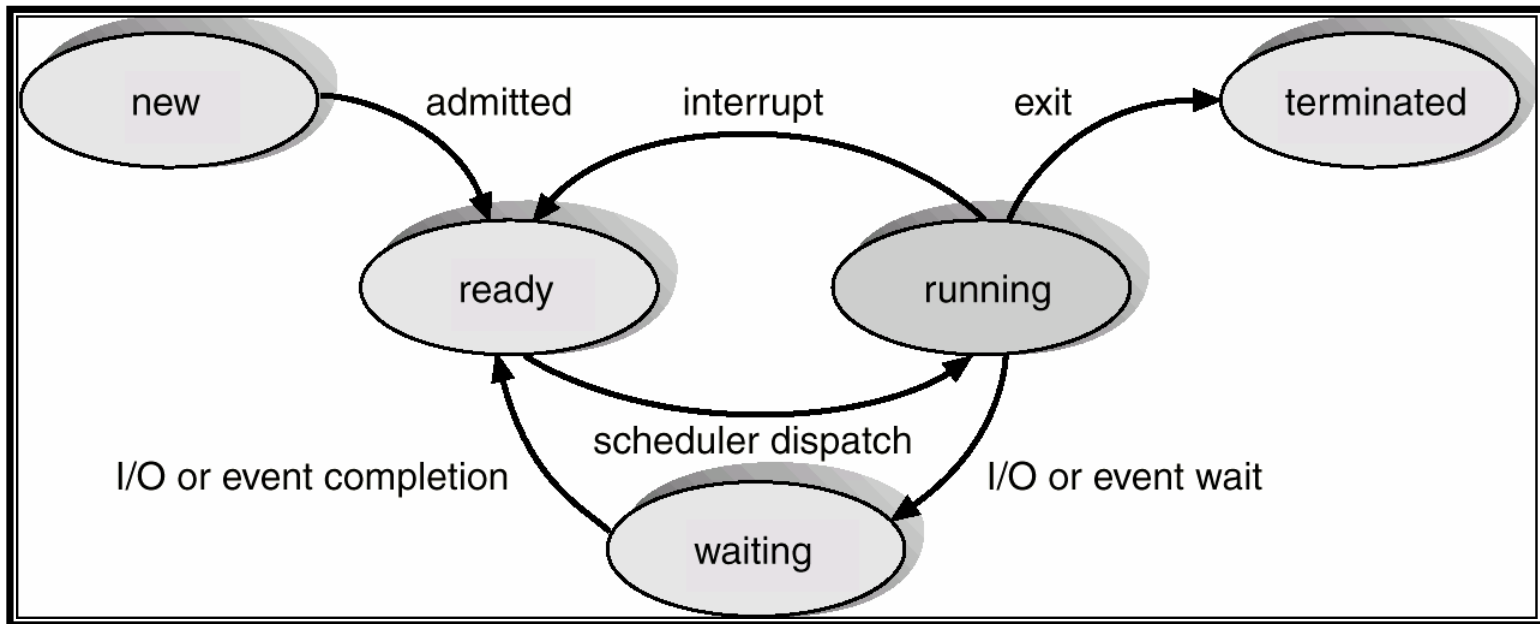
- Sistem operasi menjalankan banyak dan **beragam** program :
 - Batch system – jobs
 - Time-shared systems – user programs atau tasks
 - Istilah pada buku teks: *job*, *task* dan *process* (dapat diartikan sama)
- Proses adalah program yang dieksekusi ;
 - Aktif (proses=>memori) vs pasif (program => file)
 - Instruksi pada program (code) akan dieksekusi secara berurut (sekwensial) sesuai dengan “line code” (stored program concept).
- Proses lebih dari “program code yang aktif”:
 - Melacak posisi instruksi (sequential execution): program counter
 - Menyimpan data sementara var., parameter, return value: stack
 - Menyimpan data (initial, global variable dll): data section
 - Menyimpan status proses (contoh, aktif, wait I/O request dll.)



Status Proses

- Saat-saat proses dijalankan (executed) maka status dari proses akan berubah
 - Status proses tidak selamanya aktif menggunakan CPU).
 - Sering proses menunggu I/O complete => status wait, sebaiknya CPU diberikan kepada proses yang lain.
 - Mendukung multi-tasking – utilisasi CPU dan I/O
- Status proses (antara lain):
 - **new**: proses dibuat.
 - **running**: instruksi dieksekusi.
 - **waiting**: proses menunggu beberapa event yang akan terjadi
 - **ready**: proses menunggu jatah waktu dari prosessor
 - **terminated**: proses selesai dieksekusi.

Diagram Status Proses





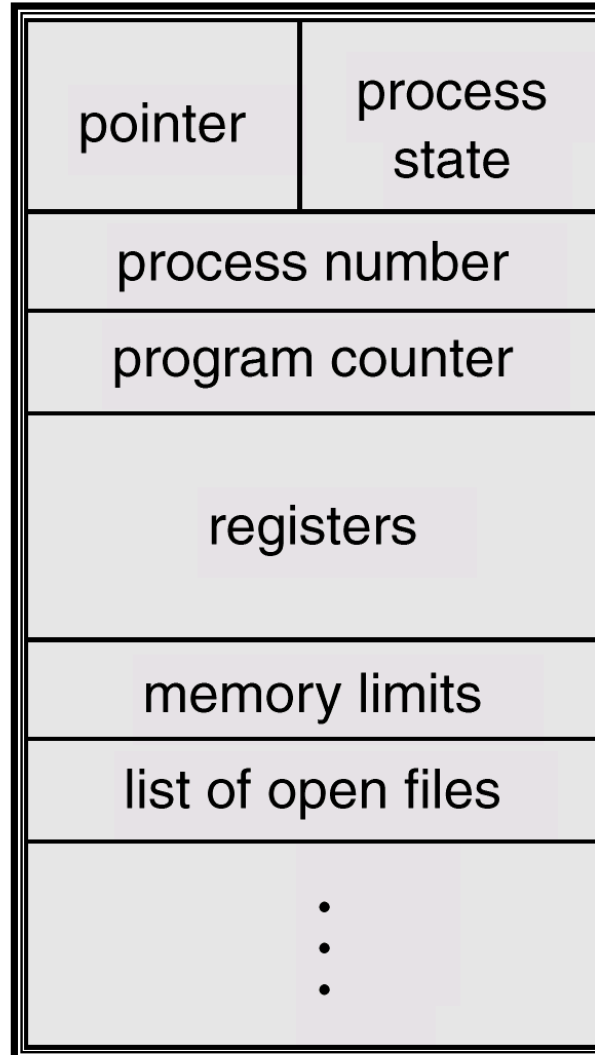
Informasi Proses

Dimanakah informasi proses disimpan?

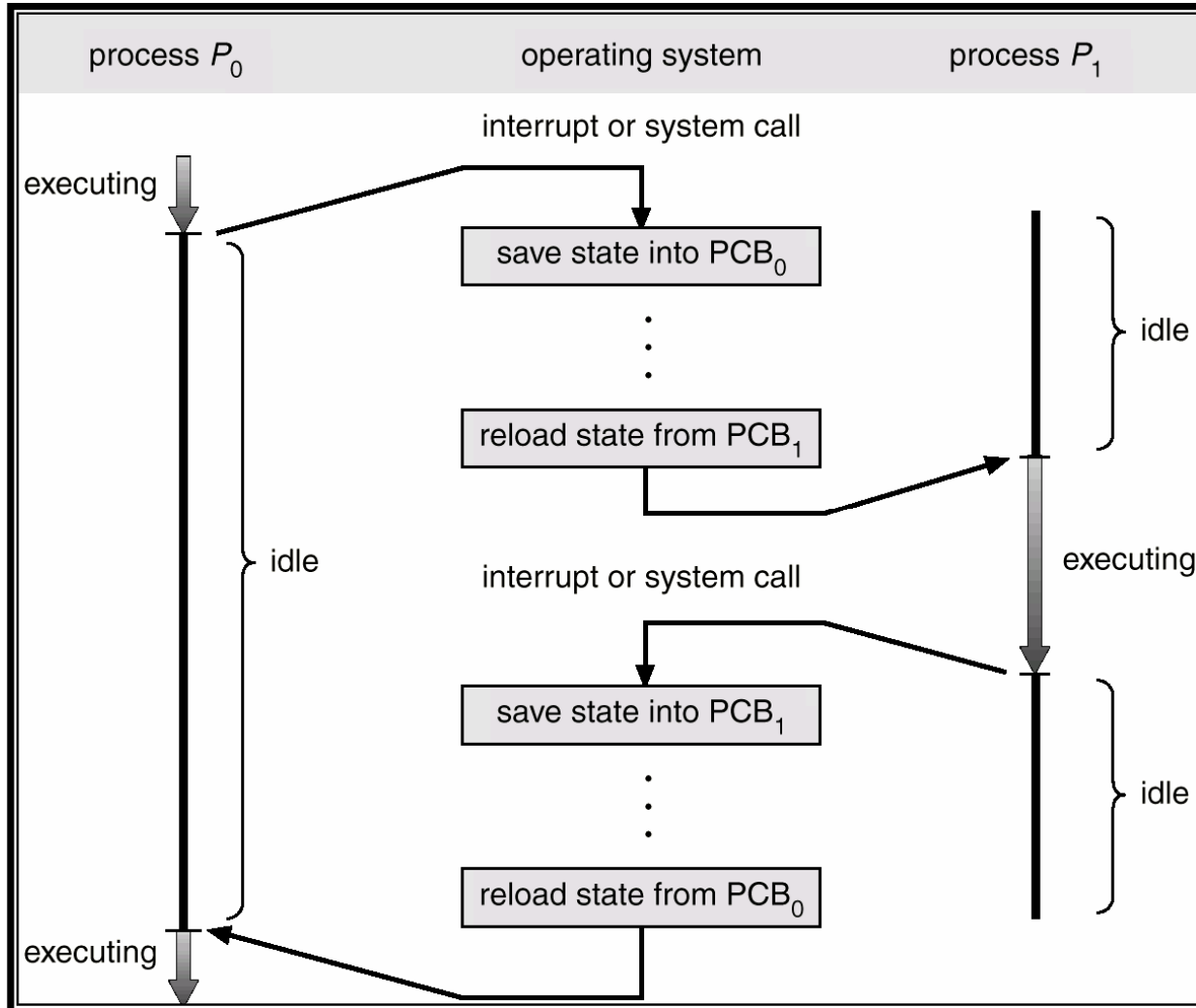
- Data struktur dari OS dalam bentuk table :
 - Satu entry table/linked list => struktur data untuk menampung informasi satu proses (array of structure).
 - Setiap entry pada tabel proses menyimpan satu proses. Contoh: MINIX (src/kernel/proc.h) => `struct proc { ... };`
- Informasi yang disimpan:
 - Informasi internal CPU: isi register-register, program counter, status CPU dll (umumnya dalam bentuk stack frame).
 - Identifikasi proses: nama proses, proses number/index, proses id.
 - Identifikasi proses: nama proses, proses number/index, proses id.
 - Accounting dan timer: user time, system time, alarm etc.
 - Resources: memory & file management.



Process Control Block (PCB)



CPU Switch Dari Satu Proses ke Proses Lainnya

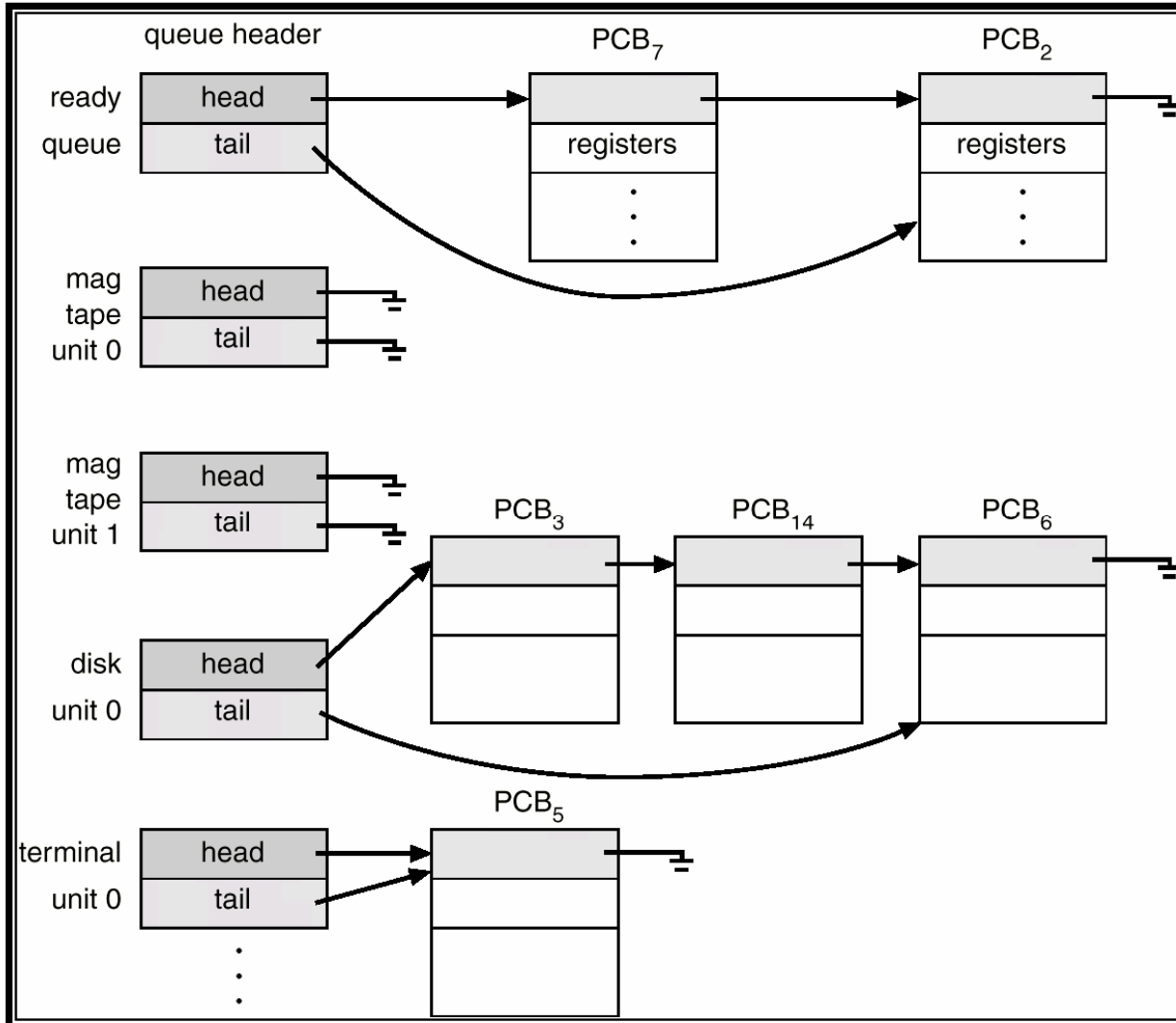


Penjadualan Proses



- Apakah tujuan dari multiprogramming?
 - “Maximize” pemakaian CPU secara efisien (jadwal dan giliran pemakaian CPU).
=> CPU digunakan oleh proses-proses terus menerus
- Apakah tujuan dari “time-sharing”?
 - Pemakaian CPU dapat di switch dari satu proses ke proses lain (concurrent process execution)
=> sesering mungkin, user dapat berinteraksi dengan sistim
- Bagaimana jika sistim prosesor tunggal?
 - “Hanya ada satu proses yang dapat dijalankan”
 - Proses lain menunggu sampai CPU dapat dijadwalkan (schedule) ke proses tsb

Ready Queue dan I/O Device Queues



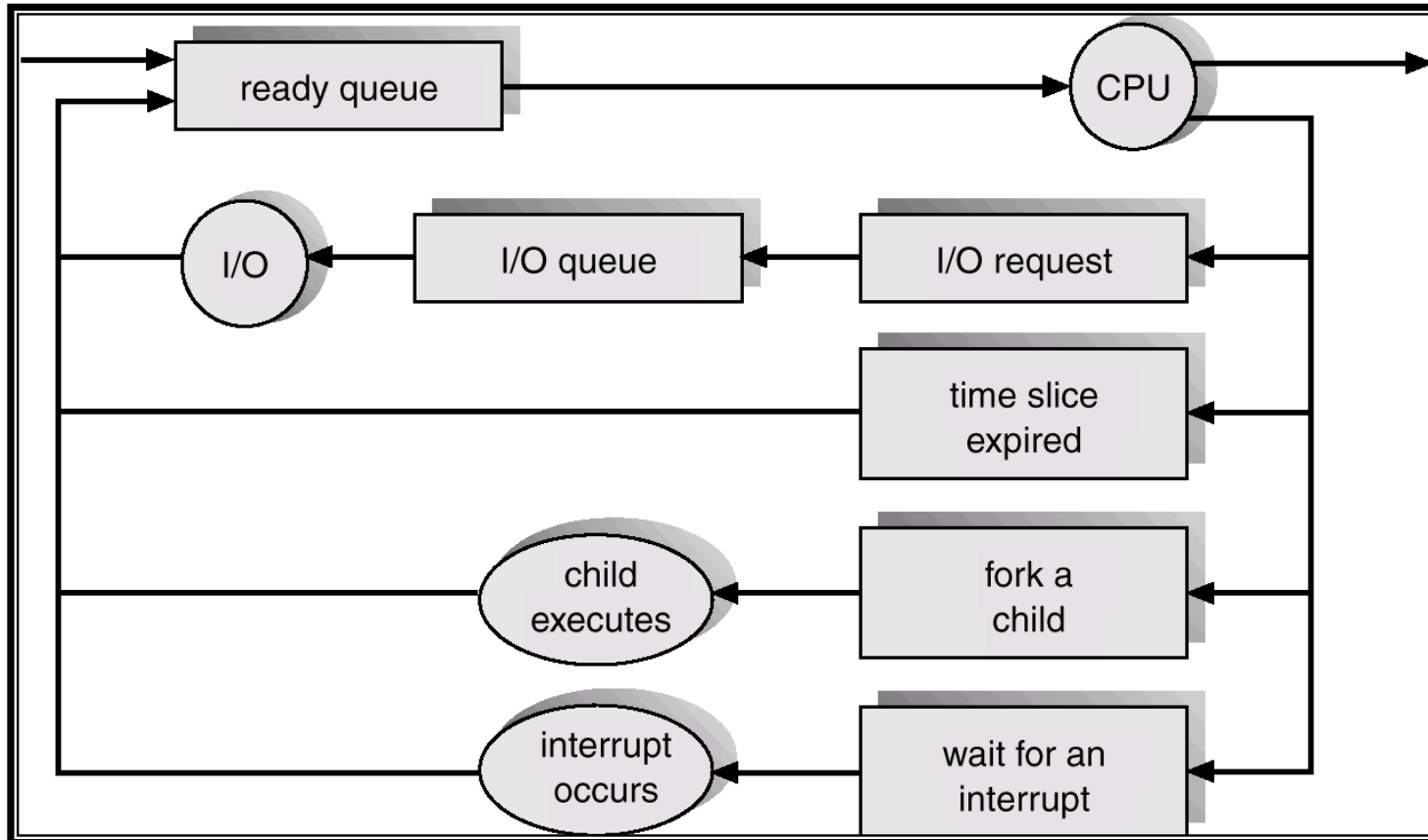
Penjadualan Proses



- Proses dapat berubah status dan berpindah dari satu antrian ke antrian yang lain
 - Proses dengan status “ready” berada di ReadyQueue
 - Menunggu giliran/dipilih oleh scheduler => menggunakan CPU
 - Selama eksekusi (status “run”) events yang dapat terjadi:
 - I/O request => I/O wait berada pada DeviceQueue
 - Create “child” proses => Jalankan proses “child”, tunggu sampai proses selesai (wait)
 - Time slice expired => Waktu pemakaian CPU habis, interrupt oleh scheduler, proses akan berpindah ke ReadyQueue



Representasi Penjadualan Proses



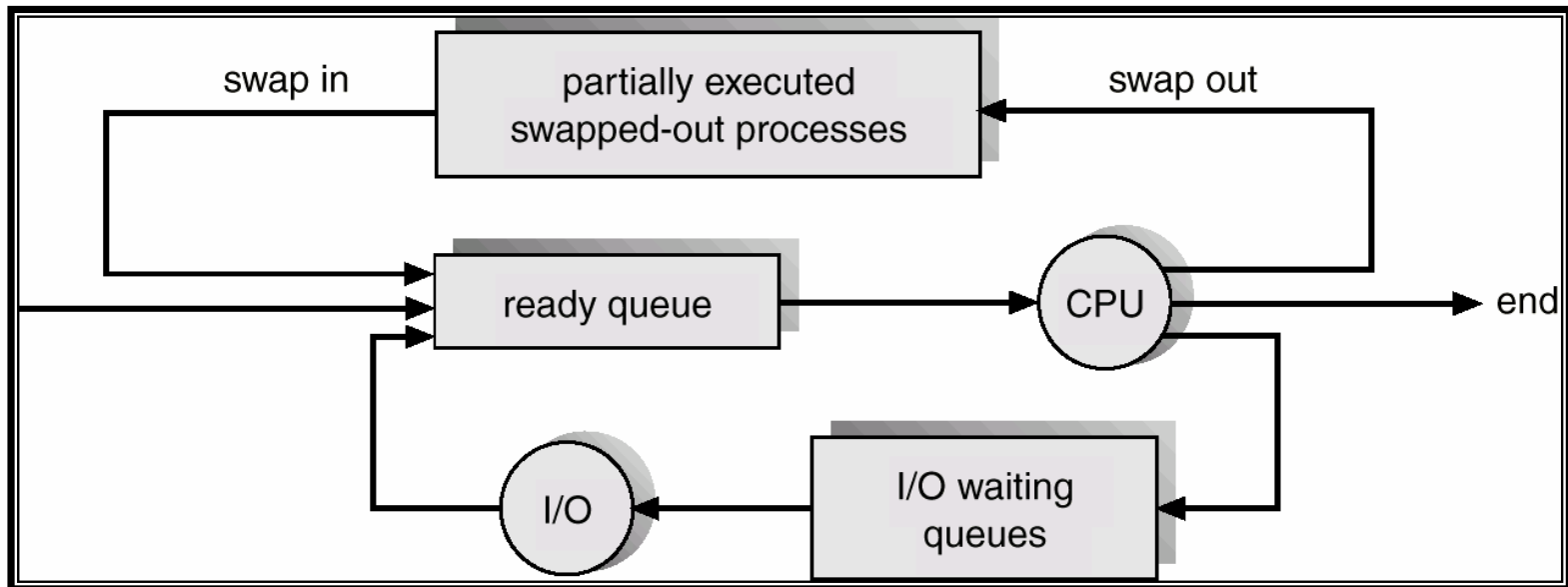


Penjadual / Schedulers

- Bagaimana schedulers memilih proses atau program (decision)?
 - Lebih dari satu proses atau program yang akan dijalankan?
- Long-term scheduler (or job scheduler) – memilih proses/program yang mana yang akan di load dan berada di **ready queue**.
 - Kemungkinan terdapat proses atau job baru.
 - Kemungkinan proses dipindahkan dari memori ke disk (swap out).
- Short-term scheduler (or CPU scheduler) – memilih proses yang mana yang berada di **ready queue** akan “run” (mendapatkan jatah CPU).



Penjadualan Jangka Menengah



Penjadual / Schedulers (Cont.)



- Long-term scheduler tidak sering (proses baru) (seconds, minutes) => (may be slow).
 - The long-term scheduler controls the *degree of multiprogramming* => *berapa banyak proses yang dapat aktif (berada di memori)*
- Short-term scheduler dijalankan sangat sering (milliseconds) => giliran pemakaian CPU dari proses- proses yang siap
 - Pada saat terjadi penggantian alokasi CPU dari satu proses ke proses lain:
 - Menyimpan informasi internal CPU dari proses yang akan digantikan (SAVE).
 - Meload kembali informasi internal CPU dari proses yang akan menggantikan.
 - Dikenal dengan istilah: context switch proses.

Alih Konteks / Context Switch



- Jika Scheduler switch ke proses lain, maka sistem harus menyimpan “informasi” proses sekarang (supaya dapat dijalankan kembali)
- Load “informasi” dari proses baru yang berada di PCB
- Waktu Context-switch adalah overhead; sistem tidak melakukan pekerjaan saat terjadi switch.
 - Sangat tergantung pada waktu di hardware
 - OS modern mencari solusi untuk mengurangi overhead waktu switch proses



Pembuatan Proses

- Umumnya proses dapat membuat proses baru (child process).
 - Child process dapat membuat proses baru.
 - Terbentuk “tree” dari proses.
- Pilihan hubungan antara parent dan child proses:
 - Resource sharing
 - Parent dan child berbagi resource
 - Children berbagi subset dari resource milik parents.
 - Parent dan child tidak berbagi resource.
 - Execution
 - Parent dan children melakukan eksekusi secara serempak.
 - Parent menunggu hingga children selesai.

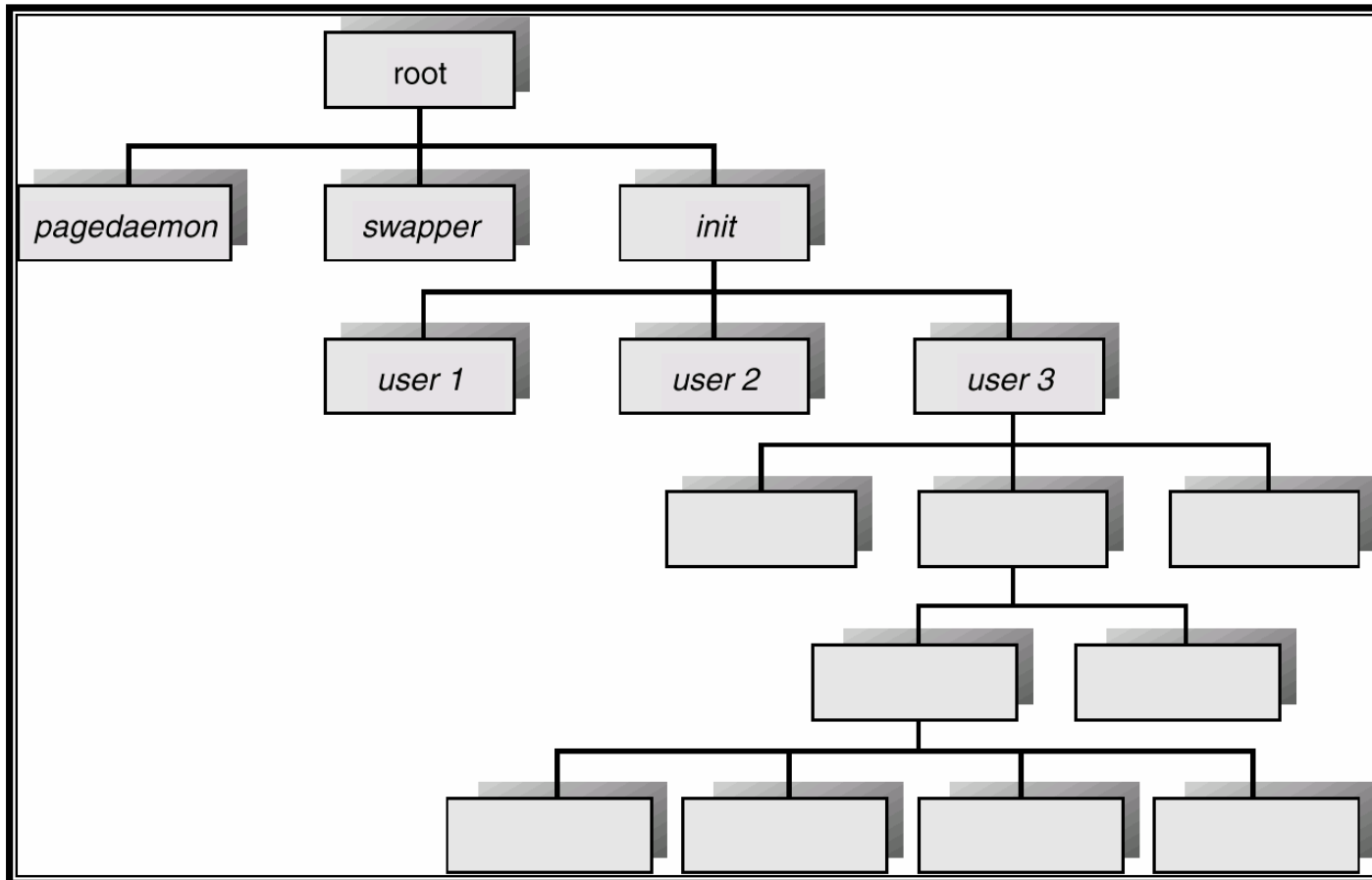


Pembuatan Proses (Cont.)

- Address space
 - Child menduplikasi parent.
 - Child memiliki program yang di load ke dalamnya.
- Contoh UNIX :
 - **fork** system call membuat proses baru
 - **execve (EXEC)** :
 - menjalankan program spesifik yang lain
 - nama program tersebut menjadi parameter dari system call
 - EXEC (sering di load sesudah menjalankan fork).
 - Tahapan pembuatan proses baru:
 - Periksa apakah masih terdapat ruang pada PCB.
 - Mencoba mengalokasikan memori untuk proses baru.
 - Mengisi informasi untuk proses baru: nama proses, id, copy data dari parent dll.
 - Mencantumkan informasi proses ke kernel OS.



Proses Tree pada Sistem UNIX





Terminasi Proses

- Proses dapat berakhir:
 - Eksekusi instruksi terakhir (atau keluar: `exit` system call).
 - OS yang akan melakukan dealokasi (memory, file resources).
- UNIX (MINIX):
 - Output signal dari child ke parent
 - Jika parent tidak menunggu (via **wait system call**), proses akan terminate tapi belum di release dari PCB (status: ZOMBIE).
 - Proses dengan status ZOMBIE (parent telah terminate), akan menjadi child dari proses "init".
- Parent dapat menghentikan eksekusi proses child secara paksa.
 - Parent dapat mengirim signal (**abort, kill system call**).



Kerjasama Proses

- Proses independent tidak mempengaruhi eksekusi proses yang lain
- Kerjasama proses dapat mempengaruhi atau dipengaruhi oleh eksekusi proses yang lain
- Keuntungan kerjasama proses :
 - Sharing informasi
 - Meningkatkan kecepatan komputasi
 - Modularitas
 - Kemudahan



Interprocess Communication (IPC)

- Mekanisme proses untuk komunikasi dan sinkronisasi aksi
- Sistem Pesan – komunikasi proses satu dengan yang lain dapat dilakukan tanpa perlu pembagian data.
- IPC menyediakan dua operasi :
 - **send**(*message*) – pesan berukuran pasti atau variabel
 - **receive**(*message*)
- Jika P dan Q melakukan komunikasi, maka keduanya memerlukan :
 - Membangun jalur komunikasi diantara keduanya
 - Melakukan pertukaran pesan melalui send/receive
- Implementasi jalur komunikasi
 - physical (shared memory, hardware bus)
 - logical (logical properties)



Komunikasi Langsung

- Proses harus diberi nama secara jelas :
 - **send** ($P, message$) – kirim pesan ke proses P
 - **receive**($Q, message$) – terima pesan dari proses Q
- Properti jalur komunikasi
 - Jalur dibangun secara otomatis
 - Setiap jalur memiliki pasangan masing-masing dalam proses komunikasi
 - Jalur komunikasi tersebut biasanya directional



Komunikasi Tidak Langsung

- Pesan dikirim dan diterima melalui mailboxes (yang ditunjuk sebagai port)
 - Proses
 - Processes can communicate only if they share a mailbox.
- Properti jalur komunikasi
 - Jalur komunikasi hanya dibangun jika proses di-share dalam mailbox
 - Jalur merupakan gabungan beberapa proses
 - Setiap pasangan proses dibagi ke dalam beberapa jalur komunikasi.



Komunikasi Tidak Langsung

- Operasi
 - Membuat mailbox baru
 - Mengirim dan menerima pesan melalui mailbox
 - Menghapus/memusnahkan mailbox
- Primitive didefinisikan :
 - send**(A , $message$) – kirim pesan ke mailbox A
 - receive**(A , $message$) – terima pesan dari mailbox A

Komunikasi Tidak Langsung



- Mailbox sharing
 - P_1 , P_2 , dan P_3 berbagi (share) mailbox A.
 - P_1 , send; P_2 and P_3 receive.
 - Siapa yang mendapat pesan ?
- Solusi
 - Memperbolehkan suatu jalur yang merupakan gabungan lebih dari dua proses
 - Hanya meperbolehkan satu proses pada suatu waktu untuk mengeksekusi operasi receive .
 - Memperbolehkan sistem untuk memilih receiver. Sender diberitahu siapa yang menjadi receiver.

Sinkronisasi



- Pesan yang disampaikan dapat di blok atau tidak (non-blocking)
- **Blocking** dikenal dengan synchronous.
- **Non-blocking** dikenal dengan **asynchronous**

Buffering



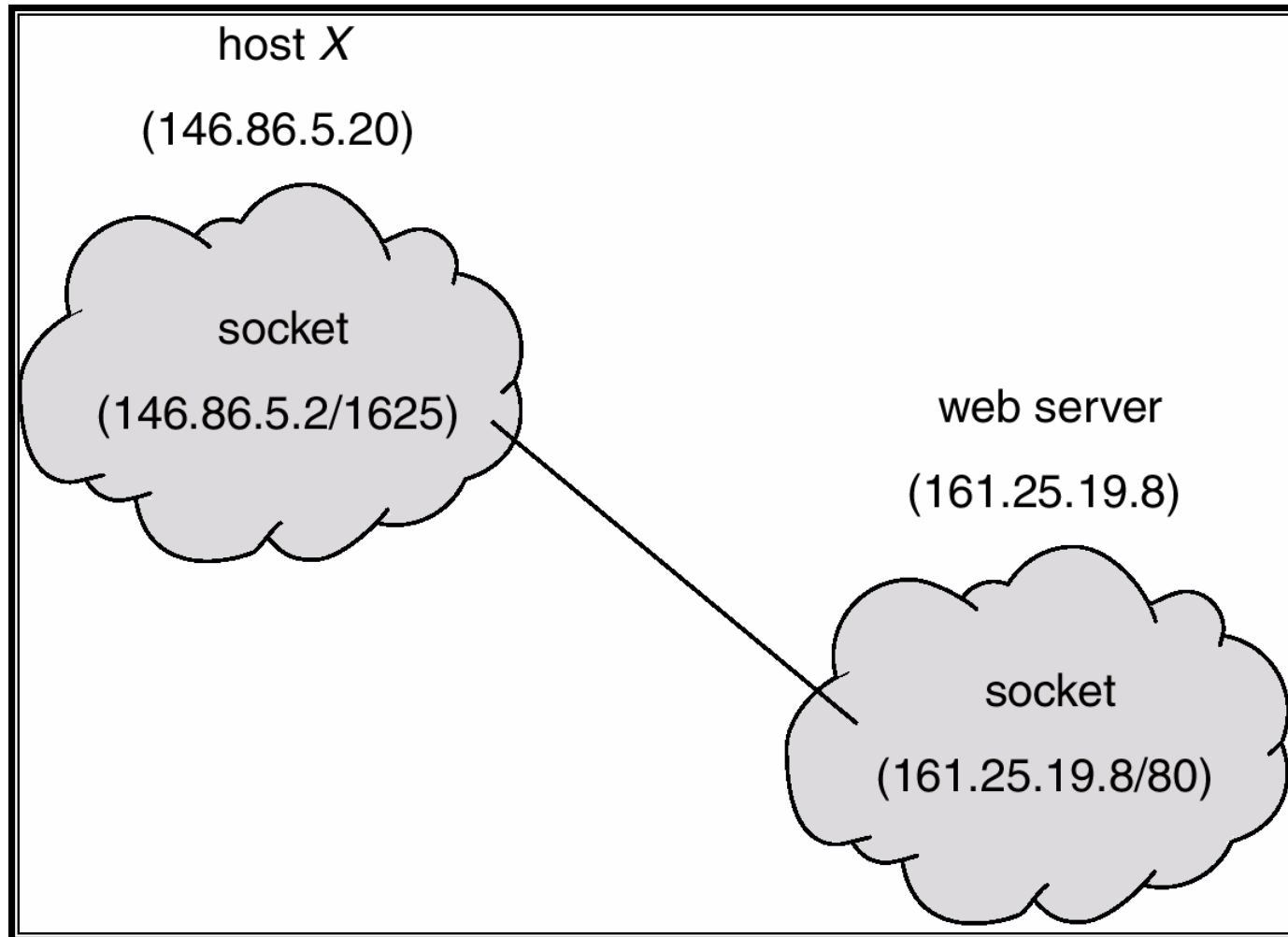
- Antrian pesan yang dihubungkan dalam suatu jalur, diimplementasikan dengan tiga jalan :
 1. Zero capacity – tidak ada pesan
 - Sender harus menunggu receiver (rendezvous).
 2. Bounded capacity – memiliki panjang yang terbatas (finite length) dari n pesan.
 - Sender menunggu pada saat jalur penuh.
 3. Unbounded capacity – memiliki panjang tidak terbatas (infinite length)
 - Sender tidak pernah menunggu.



Sockets

- Suatu socket didefinisikan sebagai titik akhir (endpoint) komunikasi
- A socket is defined as an *endpoint for communication*.
- Gabungan IP address dan port
- Socket **161.25.19.8:1625** mengacu pada port **1625** pada host **161.25.19.8**
- Komunikasi berada diantara pasangan socket

Komunikasi Socket



Remote Procedure Calls (RPC)



- Remote Procedure Call (RPC) adalah abstraksi pemanggilan prosedur diantara proses pada sistem jaringan
- **Stubs** – proxy sisi client untuk prosedur aktual pada server
- Stub sisi client ditempatkan di server dengan parameter *marshalls*.
- Stub sisi server menerima pesan, membongkarnya dengan parameter marshall dan menjalankan prosedur pada server.

Eksekusi RPC

